CS230

Small Vessel Detection in Satellite Imagery

Richard Correro
Department of Statistics
Stanford University
`rcorrero@stanford.edu`

Abstract

I develop a computer vision algorithm to detect small vessels in satellite imagery. Using an InceptionV3 architecture, this model is trained using a dataset containing approximately 193,000 satellite images, each labeled according to the presence or absence of vessels within the image. After training this model for 79 epochs, I test its performance on a dataset containing labeled satellite images. These images were captured by satellites with lower resolution imaging sensors. I evaluate the performance of the algorithm on this test data, and compare the model's accuracy to my own for this task, a proxy for the Bayes error rate.

1 Introduction

Illegal, Unreported, and Unregulated (IUU) fishing poses an existential risk for the future of our planet's fisheries. Although national legislation and international accords have been created to limit IUU activity in commercial fishing fleets, where vessels tend to be larger, small vessels operated by independent (sometimes referred to as "artisanal") fishers are often not subject to the more stringent regulations which the commercial fishers face [1]. And even when small-scale fishers are subject to regulation, enforcement is often difficult because small vessels are extremely hard to identify through traditional methods such as the use of observer ships.

The greatest obstacle to monitoring the behavior of these small vessels, however, is that they do not wish to make their actions known when they are engaging in illegal behavior. Therefore these vessels are unlikely to use systems such as Automatic Identification System (AIS), the tracking system which all large ships are required by international law to utilize, because it broadcasts sensitive information including location data. We therefore need remote, passive data sources to monitor their behavior, and one such source is satellite imagery.

Recently several new satellite imagery providers have entered the market, making satellite imagery more accessible and less expensive than ever. New providers offer greater global coverage with lower return times (the gap between consecutive imaging of any location). Although these new data sources open up many new possibilities in geospatial analysis, one major limitation is resolution. Because data providers, such as Planet Labs, inc., utilize smaller "dove" satellites for low latency imagery, image resolution is often relatively low compared to images produced by larger imaging satellites. This low resolution makes identification of small vessels extremely difficult. But the smaller vessels are also those most likely to engage in IUU activity, making accurate identification of these ships extremely important for regulators and environmental organizations [1]. It is vital therefore to use the best tools and techniques now available to build a successful small-vessel detection system for use with low resolution satellite imagery.

Problem

With the goal of small vessel detection in mind, I designed and trained a convolutional neural network model to solve the following problem: given an input satellite image of the earth's surface, which may or may not contain a vessel, determine whether there is at least one vessel present in the image.

2 Related work

The problem of vessel detection in satellite imagery has been the subject of much research for at least the last 40 years [2]. Kanjir, et. al. present a survey of many past publications regarding methods for automatic ship detection in images and synthetic aperture radar (SAR) data [2]. The many methods developed to solve this task can be broadly divided into two categories: those using heuristic methods along with shallow classification algorithms to identify vessels, and those using deep learning approaches [2].

Shallow Classification Models

Most early vessel detection methods relied primarily on heuristic methods and hand-engineered features to identify vessels [2]. Traditional algorithms from the field of computer vision would be modified to generate hand-engineered features relevant to the vessel detection task [2]. These features would be generated from each input image and then fed into a shallow classification model. This model would be trained to discriminate between vessels and non-vessels.

Yin, et. al. develop a method which is typical of this type of approach [3]. Their method uses a "modified saliency fusion algorithm" to extract segments of images which are identified as potentially containing vessels. These candidates are then filtered according to their shape - those that do not resemble the target shape, meant to approximate the shape of a vessel, are rejected. A feature vector is then generating from the image segment corresponding to each of the surviving candidates, which is then fed to a linear Support Vector Machine for discrimination.

Xia, et. al. use geometrical features, including length, width, compactness, etc. extracted from targets identified using a "dynamic fusion algorithm" [4]. They then use a Support Vector Machine as a discriminator, but unlike Yin, et. al., use a non-linear kernel.

Arguedas presents an approach to identify and classify vessels into one of four different categories using "Local Binary Patterns" [5]. Target ships are divided into three segments from which these features are extracted. The resulting features are then provided to a linear classifier.

Deep Learning Models

Deep-learning has revolutionized many areas of the field of computer vision, and vessel detection is no exception. Several papers have been published recently describing deep-learning approaches in this area.

Gallego, et. al. present a method utilizing convolutional neural networks along with a non-parametric approach, K-Nearest Neighbors, for vessel detection [6]. Following their method, images are input into the neural network which generates "neural codes" – activations of the final hidden layer – which are then stored and used in the K-Nearest Neighbors algorithm to classify new images.

Park, et. al. train a convolutional neural network to identify pair trawlers in and around the North Korean Exclusive Economic Zone [7]. They acquired PlanetScene imagery of these waters and labeled pairs of trawler vessels in the imagery. These vessels almost always fish in pairs: Park et. al. therefore trained their model to identify pairs of vessels rather than individual ships [7]. As pairs of vessels present a much larger feature than a single ship, this allowed them to achieve better performance than otherwise possible by identifying each vessel individually. This approach is very similar to the approach developed in this paper, except here we are interested in identifying all classes of vessels, particularly small vessels, not just pair trawlers.

Although many approaches have been developed in this area, and, according to the authors, many have achieved great performance, there is one bias which makes previous approaches sub-optimal for the task of identifying vessels involved in IUU activity. All of these past approaches focus more on larger vessels, to the detriment of model performance on smaller vessels. Smaller vessels are much harder to identify accurately and precisely, and the economic incentives to develop systems to identify small vessels are nowhere near as great as the incentives to identify larger vessels. The clear theme of most past research is that small vessels have been marginalized, while larger vessels receive the lion-share of the attention. Since small vessels are the most likely to participate in IUU activity, this is a major problem [1].

3 Dataset and Features

Because of the large number of training samples required to train a convolutional neural network from scratch, I obtained images from two openly-available datasets [6][8]. Each dataset contained images from satellite photographs and labels denoting the presence or absence of vessels in each image. Although each dataset included satellite images obtained from multiple classes of satellites, there were slight differences in the coverage of each. The smaller dataset contained more images including smaller vessels, but because this dataset was approximately 33 times smaller than the larger dataset, I chose to combine them into a single training dataset [6]. This dataset contained 193,000 samples. From this set I created a validation by selecting 1,930 samples. I chose these samples so that the class distributions in the validation set and the training set were equal, each containing approximately 91.98 percent negative samples, and 8.02 percent positive samples.

Although the size of training images varied, the most common size was 768 by 768 (with three channels). The InceptionV3 architecture, however, expects input images of size 299 by 299 by 3 [9]. I therefore down-sampled the training image to a standard size of 299 by 299 by 3, with an interpolation of 2.0.

Data Transformation

To augment the training set, two transformations were randomly applied to each sample dynamically immediately before input into the model: each image was first randomly flipped horizontally (with probability 0.5), then randomly flipped vertically (also with probability 0.5).

Because the satellite imagery contained in the training dataset was acquired by larger, more traditional imaging satellites, the spatial resolution of these images is generally greater than that of the target imagery [6][8]. To approximate the lower spatial resolution of the target imagery, a Gaussian blur filter with a radius of 2.0 was randomly applied to each training image on input. The probability blurring was set at 0.5 for the first 13 epochs, and at 0.85 for the last 66 epochs. For the validation set, all samples were blurred (see Figures 1 and 2).

Although the target imagery is generally of lower spatial resolution than the training images, it is desirable that the model perform well with higher-resolution imagery as well: this is why some percentage of the samples in the training dataset on each epoch were not blurred.

Test Data

The target imagery for this model is Planet Labs, inc.'s PlanetScene satellite imagery. These images are 24 by 7 kilometer four channel (blue, green, red, near-infrared) images with a spatial resolution of 3 meters per pixel. To create a test set representative of the target distribution, I acquired approximately 300 PlanetScene images of the littoral waters near the coast of Peru. I then divided each image into 299 by 299 by 3 tiles (the near-infrared channel was ignored since the training data includes only the three color channels). This width and height were chosen because they are the dimensions required by the InceptionV3 model.

I labeled each image according to the presence (label = 1) or absence (label = 0) of vessels within it. Because vessels are relatively sparse on open waters, I chose a distribution of 10 percent positive samples and 90 percent negative samples in the test dataset. This distribution is roughly equivalent to the class distribution of the training data. The total number of samples created was 1,926.

4 Methods

For my classification model I used the InceptionV3 architecture, implemented using Pytorch [9][10][11]. I modified the model architecture by replacing the final softmax layer with a single-unit output layer since there are only two classes in this task.

The key advantage of the Inception architecture comes from its utilization of inception blocks, allowing it to achieve near state-of-the-art performance on benchmark computer vision tasks with a model size smaller on average than architectures with similar performance [12]. The InceptionV3 model accepts input tensors of size 299 by 299 by 3.

Training

I trained the model using a cross-entropy loss function and Adam optimization with parameters

$$\begin{aligned} \alpha &= 0.00001\ (1e-4) \\ \beta_1 &= 0.9 \\ \beta_2 &= 0.999 \\ \lambda &= 0.000001\ (1e-5) \end{aligned}$$

for the first 13 epochs, where α denotes the learning rate, $\beta_1,\ \beta_2$ are the forgetting factors for gradients and second moments of gradients, respectively, and λ is the $L2$ regularization (also known as weight decay) parameter. After completion of the 13th epoch, I set

$$\lambda = 0.00000001\ (1e-7).$$

A minibatch approach was used, with a minibatch size of 64, chosen because this was the larget batch size which could be reliably loaded into the GPU's RAM.

Procedure

I trained the model for 79 epochs in total. Upon the completion of each epoch, the model was tested using the validation set for performance analysis, and running loss was calculated upon the completion of each tranche of 100 minibatches (6400 training samples).

5 Experiments/Results/Discussion

Training Performance

The model was trained for 79 complete epochs on the training dataset, seeing 15,182,640 images in total and running for 357.6 hours.

The mean loss on the validation set consistently decreased from epochs 1 - 45, after which loss remained roughly constant, hovering around 0.06 (see Figure 3). The minimum loss recorded on the validation set was 0.05, encountered after both the 63 and 77th epochs.

Mean accuracy on the validation set consistently increased for the first 50 epochs, after which it stabilized at approximately 98 percent (see Figure 4). The peak validation accuracy was 98.4 percent, recorded after the 72nd epoch.

The validation dataset contains a disproportionate number of negative samples, and therefore a naïve model trained to label all samples as negative would achieve a validation accuracy of 91.98 percent. We see that the model's accuracy after the first epoch was 92.0 percent, barely above this threshold. Model performance consistently improved thereafter.

Test Performance

After completion of training, the model was evaluated on the test data. The model achieved an average test accuracy of 97.93 percent, not far below the maximum accuracy achieved on the validation set.

The model achieved an average precision of 88.28 percent, and an average recall of 82.04 percent (see Figure 5). The model generated more false negatives than positives.

Analysis

The model achieved an average accuracy of 97.93 percent on the test data, not far below the maximum accuracy of 98.4 percent recored on the validation data.

To estimate the Bayes error rate for this task, I chose 300 images at random from the training data and labeled them according to the presence or absence of vessels. After labeling the images, I compared my labels to the ground truth labels, and found that my own accuracy was roughly 99 percent, only beating the performance of the model by approximately one percentage point.

With an average precision of 88.28 percent and an average recall of 82.04 percent, the model is more hesitant to generate positive labels than negative labels for samples about which it is unsure. Because the model is intended to be used on imagery of open oceans, where vessels are sparse and relatively uncommon in satellite images, this is a desirable trait. If the false positive rate is too high, then false positives will dominate the true positives in practice.

6 Conclusion/Future Work

I developed a computer vision algorithm to detect small vessels. Using an InceptionV3 architecture, this model was trained using a dataset containing approximately 193,000 satellite images, each labeled according to the presence or absence of vessels within the image. After training this model for 79 epochs, I tested its performance on a dataset I created containing labeled satellite images. These images were captured by a lower resolution imaging sensor. I evaluated the performance of the algorithm on this test data, and compared the model's accuracy to my own for this task, a proxy for the Bayes error rate. The model achieved an accuracy within one percentage point of my own.

Future Work

Obvious extensions of this work include the creation of an object detection model which not only labels images according to the presence or absence of vessels, but also generates probabilistic bounding boxes demarcating the location of vessels in the images. Such a model could use the hidden layers of the InceptionV3 model as a "backbone", transforming input images before feeding the activations of the final hidden layer into the object detection model [13]. Alternatively, this classification model could be used to identify images likely to contain vessels. Once these images are identified, a separate object detection model could be used to label these likely candidates. This would be useful because the object detection model would likely be much more computationally-expensive to run, and as most satellite images will not contain vessels, performing object detection on every sample is unnecessary in practice.

7 Contributions

This project was entirely my own work. I received guidance from researchers affiliated the Center for Ocean Solutions at Stanford, were I am currently conducting research. The results of this project will be used in future research.

References

[1] Fréon, Pierre, et al. "Environmentally extended comparison table of large-versus small-and medium-scale fisheries: the case of the Peruvian anchoveta fleet." Canadian journal of fisheries and aquatic sciences 71.10 (2014): 1459-1474.

[2] Kanjir, Urška, Harm Greidanus, and Krištof Oštir. "Vessel detection and classification from spaceborne optical images: A literature survey." Remote sensing of environment 207 (2018): 1-26.

[3] Y. Yin, N. Liu, C. Li, W. Wan and T. Fang, "Coarse-to-fine ship detection using visual saliency fusion and feature encoding for optical satellite images," 2016 International Conference on Audio, Language and Image Processing (ICALIP), Shanghai, 2016, pp. 705-710, doi: 10.1109/ICALIP.2016.7846579.

[4] Y. Xia, S. Wan and L. Yue, "A Novel Algorithm for Ship Detection Based on Dynamic Fusion Model of Multi-feature and Support Vector Machine," 2011 Sixth International Conference on Image and Graphics, Hefei, Anhui, 2011, pp. 521-526, doi: 10.1109/ICIG.2011.147.

[5] V. F. Arguedas, "Texture-based vessel classifier for electro-optical satellite imagery," 2015 IEEE International Conference on Image Processing (ICIP), Quebec City, QC, 2015, pp. 3866-3870, doi: 10.1109/ICIP.2015.7351529.

[6] Gallego, Antonio-Javier, Antonio Pertusa, and Pablo Gil. "Automatic ship classification from optical aerial images with convolutional neural networks." Remote Sensing 10.4 (2018): 511.

[7] Park, Jaeyoon, et al. "Illuminating dark fishing fleets in North Korea." Science advances 6.30 (2020): eabb1197.

[8] Faudi, Jeff, "Airbus Ship Detection Challenge", Version 2 (2018). Retrieved September 30, 2020 from https://www.kaggle.com/c/airbus-ship-detection/overview.

[9] Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." Advances in neural information processing systems. 2019.

[10] Harris, Charles R., et al. "Array programming with NumPy." Nature 585.7825 (2020): 357-362.

[11] McKinney, Wes. "Data structures for statistical computing in python." Proceedings of the 9th Python in Science Conference. Vol. 445. 2010.

[12] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[13] Amjoud, Ayoub Benali, and Mustapha Amrouch. "Convolutional Neural Networks Backbones for Object Detection." International Conference on Image and Signal Processing. Springer, Cham, 2020.

Code available at https://github.com/rcorrero/poisson

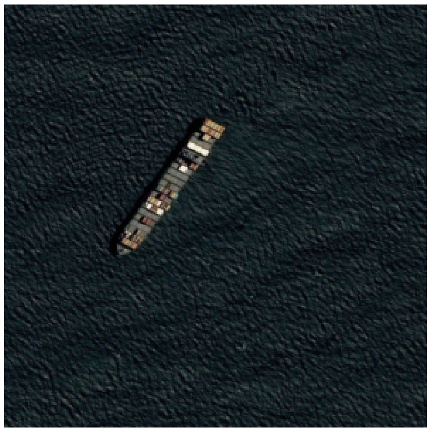


Figure 1: Original training image.

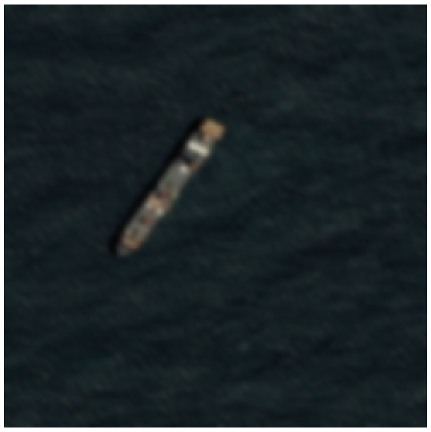


Figure 2: Effect of Gaussian blur.

Figure 3: Validation loss by epoch.

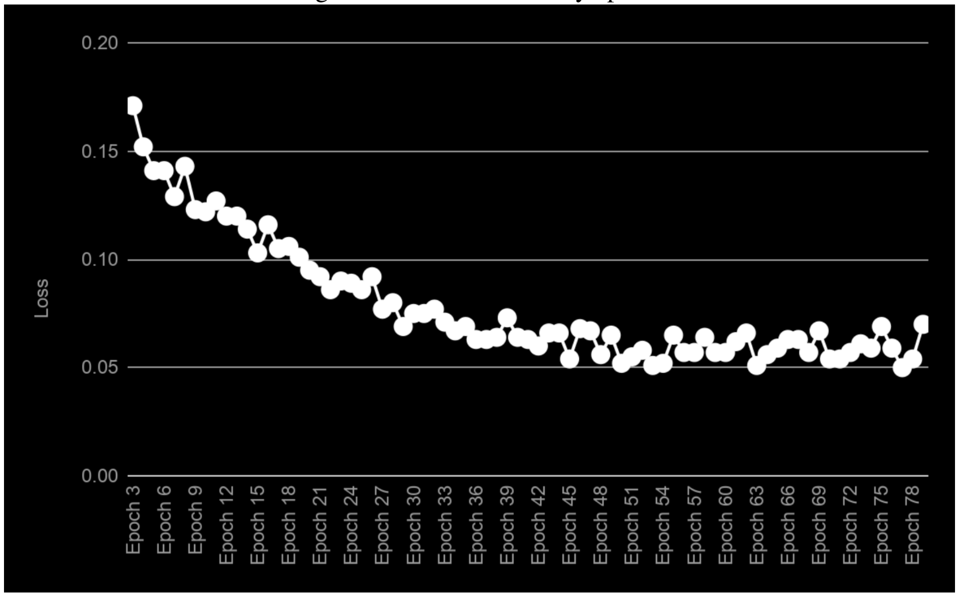


Figure 4: Validation accuracy by epoch.

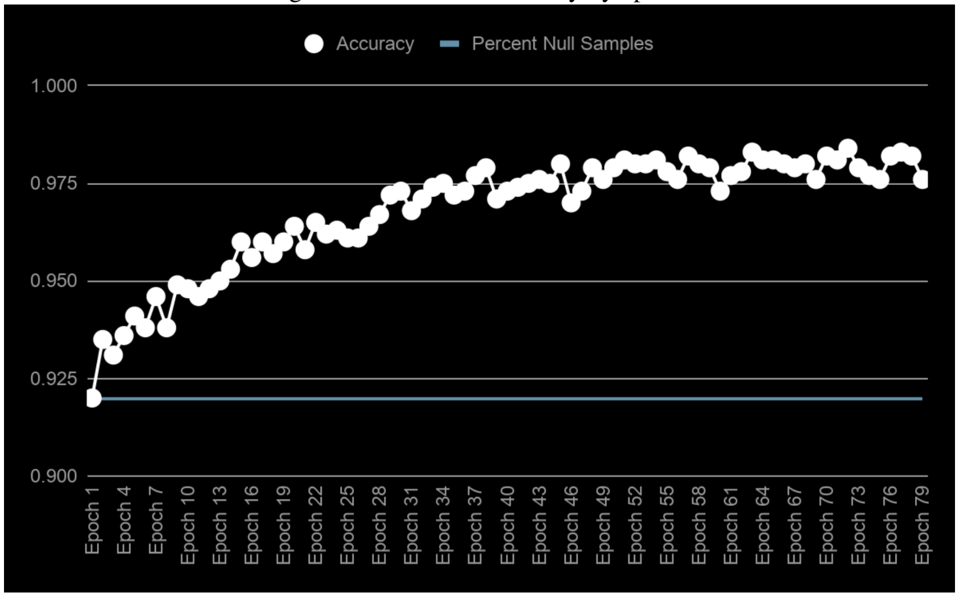
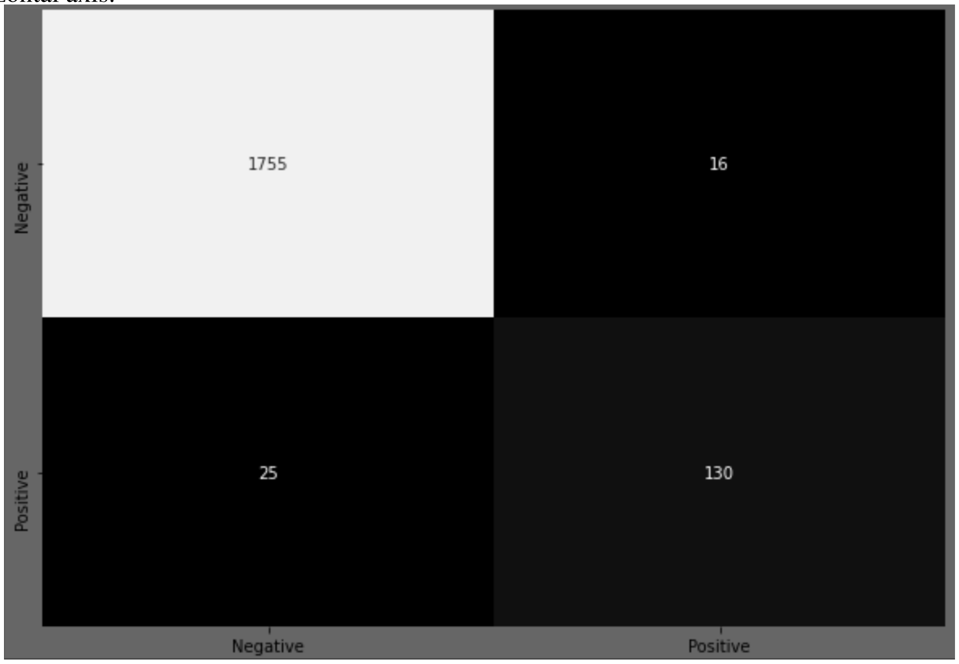


Figure 5: Confusion matrix for test data. Predicted labels along vertical axis and true labels along horizontal axis.


Negative
Positive
Negative
Positive